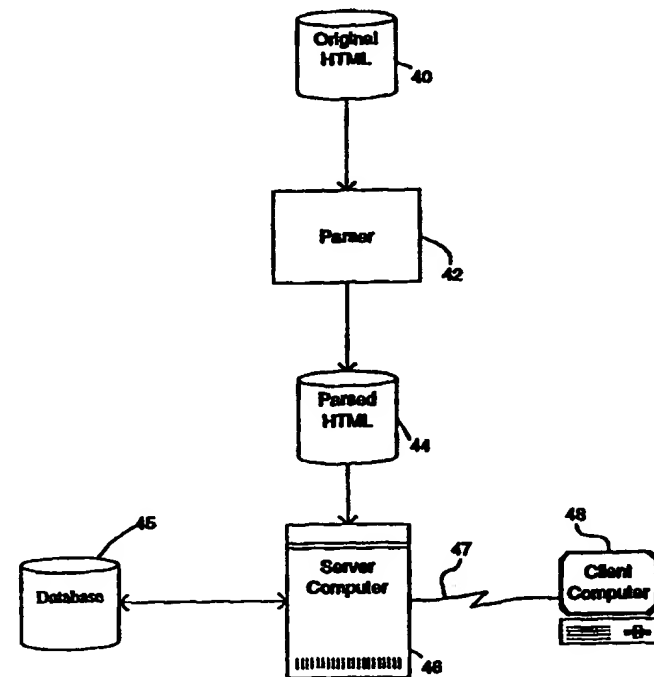


PCTWORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁶ : G06F 17/30	A1	(11) International Publication Number: WO 99/59087 (43) International Publication Date: 18 November 1999 (18.11.99)
(21) International Application Number: PCT/US99/06676 (22) International Filing Date: 26 March 1999 (26.03.99) (30) Priority Data: 09/078,084 13 May 1998 (13.05.98) US (71) Applicant: BULL HN INFORMATION SYSTEMS INC. [US/US]; 300 Concord Road, Billerica, MA 01821-4186 (US). (72) Inventor: GIROUX, Michael; 14830 North Black Canyon Highway #2114, Phoenix, AZ 85023 (US). (74) Agent: SOLAKIAN, John, S.; Bull HN Information Systems Inc., 300 Concord Road, Law Office MA30-883A, Billerica, MA 01821-4186 (US).		(81) Designated States: European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE). Published <i>With international search report.</i>
(54) Title: METHOD AND APPARATUS FOR IMPROVING CODE EXECUTION PERFORMANCE BY USING PARSED HTML (57) Abstract <p>An interpreter (42) first parses source Hyper-Text Markup Language (HTML) scripts (40) for active or dynamic Web pages into static and dynamic sequences. These static and dynamic sequences of HTML are written into variable length records in a parsed HTML file (44), with each such record consisting entirely of static or of dynamic HTML commands. A database server (46) in response to a query from a Web browser on a client computer (48) reads the parsed HTML (44), transmitting (47) static HTML records without interpretation to the browser (48), while interpreting the dynamic HTML records and performing required database (45) actions before transmitting (47) the results to the browser (48).</p> 		

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakhstan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LI	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		
EE	Estonia						

METHOD AND APPARATUS FOR IMPROVING CODE EXECUTION PERFORMANCE BY USING PARSED HTML

Field of the Invention

5 The present invention generally relates to improving performance of code execution of a Web server, and more specifically to improving code execution performance in a client/server environment by using parsed HTML source code.

Background of the Invention

10 A World Wide Web (WWW) site typically consists of a collection of HyperText Markup Language (HTML) documents. HTML is a text language that provides for hyper-linked graphic display. A user of a Web browser utilizing the World Wide Web (WWW) typically requests that a Web server download HTML text to his Web browser. The Web browser interprets the downloaded HTML text and generates screen images from the
15 HTML text. The HTML text invariably describes hyper-linked hot spots, that cause further downloads when selected.

 HTML documents are typically generated by text editors or by more specialized HTML document editors and are stored as text files in directories on Web servers. By convention, HTML text files have file
20 names that include extensions of either ".HTML" or ".HTM". Web servers recognize these file extensions and treat such documents as static byte streams. Static HTML files are typically transmitted verbatim by Web servers to Web browsers, and are thus fairly efficient for the Web servers to process.

25 One recent application that has gained popularity on the World Wide Web is database access. HTML provides an efficient, flexible method of providing a sophisticated user interface to databases. Many WWW access requests ultimately are translated into database accesses. Part of the

flexibility of using HTML for this type of application is that user interface changes tend to be reasonably easy and do not require the significant programming resources that were required by earlier generations of database interfaces.

5 To support dynamic or customized Web content, modern commercial Web servers typically recognize additional file types. For example, Microsoft recognizes files with file extensions of ".ASP" indicating that the file is an "Active Server Page" containing various fields that must be analyzed by a Web server. In some instances, these various active or
10 dynamic fields are embedded database requests.

 Unfortunately, one problem that frequently arises when utilizing active or dynamic server pages is that having a Web server interpret each HTML command in a dynamic HTML file typically requires a significant amount of computer resources. The present solution is to scale the server up
15 by replicating the database and database servers to the extent necessary to provide required levels of service. Currently, some applications are implemented with databases and database servers replicated upwards of thirty times, with access to the replicated servers provided by sophisticated high speed load leveling routers.

20 This approach works reasonably well for fairly small databases since the amount of data that needs to be replicated for each database server is fairly small. However, this solution does not scale well. In particular, this approach is currently totally infeasible for enterprise level databases consisting of terabytes of data. One reason for this infeasibility is that large
25 companies often have a hard enough time keeping online access to a single copy of their enterprise level databases, given the size of these databases. Replicating the databases even a couple of times is not feasible. To this should be added the potential for concurrency problems between database copies that arise any time there is multiple database copies in use and the
30 difficulty of supporting online update for replicated data.

 There is therefore a need to provide an efficient mechanism for Web servers to process HTML files containing active or dynamic HTML commands. This is particularly important when the dynamic HTML

commands provide access to enterprise level databases. Any such mechanism should reduce the overhead for interpreting active HTML scripts in these applications.

Brief Description of the Drawings

5 The features and advantages of the present invention will be more clearly understood from the following detailed description taken in conjunction with the accompanying FIGURES where like numerals refer to like and corresponding parts and in which:

10 FIG. 1 is a block diagram of a data processing system, in accordance with the present invention;

 FIG. 2 is a flowchart illustrating construction of parsed HTML file by a parser interpreting an original HTML file, as shown in FIG. 1;

 FIG. 3 is a flowchart illustrating operation of a database server interpreting parsed HTML, as shown in FIG. 1;

15 FIG. 4 is a sample sequences of original HTML;

 FIG. 5 is a sample sequence of parsed HTML (PHTML) generated from the sample HTML in FIG. 4; and

 FIG. 6 is a block diagram illustrating a General Purpose Computer.

Detailed Description

20 An interpreter first parses HyperText Markup Language (HTML) scripts for active or dynamic Web pages into static and dynamic (or active) portions. The static HTML is typically transmitted without change to Web browsers in response to WWW requests from the browsers. Dynamic or active HTML commands on the other hand provide database access
25 commands to the database servers. In response to these dynamic HTML commands, the database server will typically perform database queries, and

report the result of these queries to the requesting browser. Static and dynamic sequences of HTML are written into variable length records in a parsed HTML file. Each variable length record will preferably consist entirely of static HTML commands, or entirely of dynamic HTML

5 commands. Preferably, the dynamic HTML records and static HTML records alternate in the parsed HTML file in most instances. A database server in response to a query then reads the parsed HTML. Static HTML records are transmitted without interpretation to the Web browser, while the dynamic HTML records are interpreted, causing database queries to be
10 performed, with the results being transmitted to the Web browser. One advantage of this mechanism is that it is not necessary for a database server to ever interpret the static HTML text. This results in significant savings of computing resources.

FIG. 1 is a block diagram of a data processing system, in accordance
15 with the present invention. Original HTML text 40 stored on disk is parsed by a parser 42 into parsed HTML (PHTML) text 44, again stored in a Non-Volatile Storage Medium. The parser sequentially interprets the original HTML text 40. Sequences of static HTML commands are collected into static HTML records, and sequences of active or dynamic HTML
20 commands are collected into dynamic HTML records. These variable length static and dynamic HTML records are then written to the parsed HTML (PHTML) text 44 file stored in a Non-Volatile Storage Medium.

A web server computer 46, then communicates over a communications line 47 with Web browsers, executing on client computers
25 48. In accessing enterprise level databases, the web server 46 is typically a mainframe computer. Presently, client computers 48 are typically personal computers (PCs). However, it is envisioned that clients 48 will take other forms in the future. The Web server computer 46 is utilized to provide database access to an enterprise level database 45. A Web server 46
30 database program sequentially reads the parsed HTML (PHTML) text 44 in response to a Web request. The contents of static HTML records are transmitted without modification to the client computer 48. Dynamic HTML records contain dynamic HTML commands. These dynamic HTML commands provide database query commands, which in turn result in

database accesses to the database 45. The results of the database 45 search are then transmitted from the server computer 46 to the client computer 48 executing the Web browser.

5 A number of different formats of parsed HTML (PHTML) files 44 are envisioned by the current invention. A first file format utilizes records where a record length is followed by the record contents. The record length may be in ASCII, EBCDIC, or binary, depending on the characteristics of the database computer. In one preferred embodiment, a four-digit ASCII length precedes each record. This conforms to HTTP 1.1 chunk encoding and provides for record lengths of up-to 9999 bytes in length. File systems 10 in most computer architectures automatically identify the end-of-file. For those file systems that do not provide this capability, a zero length record can be used to identify the end of the file. A zero length record can also be transmitted from the server 46 to the client 48 when utilizing HTTP 1.1 15 chunk encoding to indicate end-of-file. The first couple of bytes are then read from each record to determine whether the record contains dynamic HTML, or static HTML.

Many computer architectures directly support variable length records. In these architectures, it would most often be preferable to utilize a standard 20 variable length record format. For example, the standard symbolic format utilized in Unisys 2200 systems is System Data Format (SDF). This format can be used directly. VAX systems by Digital Equipment Corporation support a similar variable length record format. Count key data records provide a variable record length format on International Business Machines. 25 These architectures typically provide an accurate end-of-file indication, and thus would not require the use of the zero length records to indicate such.

One alternate embodiment is to store the parsed HTML (PHTML) 44 as variable length records in a database. Instead of interpreting the first couple of bytes in each record to determine whether it contains static or 30 dynamic HTML, a flag can be included in each database record for this purpose.

Another alternate embodiment is to guarantee that static and dynamic HTML records alternate in the file. Zero length records may then be

utilized for the rare situations where this alternating does not occur naturally. The reason that this would be a rare occurrence is a result of the operation of the flowchart in FIG. 2 below. Zero length records are typically only necessary when the record length exceeds the maximum

5 supported record length. Taking this a step further, a single record descriptor could be used for alternating static and dynamic HTML text streams. Each such record descriptor would have a field for the length of the static HTML text, and a field for the length of the dynamic HTML text. For example, two four-byte record lengths could precede each double

10 record. Utilizing such an encoding mechanism, an end-of-file could be indicated with both length fields being zero. Both of these alternate embodiments have the advantage that interpreting the first couple of bytes of each HTML record is not necessary to determine whether the record is dynamic or static, since this determination can be made positionally. One

15 cost of these embodiments is slightly longer parsed HTML files 44. However, the added length in most instances is probably de minimis. The

original HTML file 40 is opened (not shown) and a loop is entered. The next HTML command is read, step 52. A test is then made whether the HTML command just read is of the same type, static or dynamic, as the

20 HTML commands being accumulated, step 54. If the type of the HTML command just read, step 52, is different from the type of HTML commands being accumulated, a record containing the HTML commands being accumulated is completed and written out to the parsed HTML file 44, step 56. A new record is then started to accumulate the type of HTML

25 commands just read, step 58. In any case, the HTML command just read, step 52, is then added to the record under construction, step 60. A test is then made whether there is any more input, step 62. If there is more input to be processed from the original HTML file 40, step 62, the loop repeats, and the next HTML command is read, step 52. Otherwise, when no more input

30 remains from the original HTML file 40 to be processed, step 62, the last record is completed and written to the parsed HTML file 44, step 64, and that file is then closed (not shown). As noted above, this mechanism for building a parsed HTML file 44 typically generates alternating static and dynamic HTML records.

FIG. 3 is a flowchart illustrating operation of a Web server 46 interpreting parsed HTML 44, as shown in FIG. 1. In response to a Web request, the Web server 46 opens the parsed HTML file 44 and enters a loop. Within the loop, the next record descriptor is read, step 72. A test is then made whether there are more records in the file, step 74. This is typically a test for end-of-file. Most modern computer architectures automatically provide such an indication. If the end-of-file has not been detected, step 76, a test is made whether the record contains dynamic HTML, step 80. In the preferred embodiment, interpreting the start of the first HTML command in the next record does this. However, other types of record formats would require correspondingly different types of testing. If the record contains dynamic HTML, the HTML in the record is processed, step 82. This typically results in database accesses in response to the HTML commands, with the results of the database queries being transmitted to the Web browser on the client computer 48. Otherwise, when static HTML is being processed, the entire record is transmitted to the Web browser executing on the client computer 48 without interpretation by the server computer 46, step 84. In either case, the loop is then repeated, starting with reading the next record descriptor, step 72. When no more records remain to be processed, step 74, end-of-file processing is performed, step 86. A record length of zero may be transmitted at this point to indicate end-of-file to the Web browser on the client system 48.

FIG. 4 is a sample sequences of original HTML 40. FIG. 5 is a sample sequence of parsed HTML (PHTML) 44 generated from the sample HTML 40 in FIG. 4. The first section of the original HTML 40 contains static HTML commands. In the parsed HTML 44, a record length of "0288" bytes is inserted before these static HTML commands. This is followed by an active or dynamic set of HTML commands. These are preceded by a length of "0059" in the parsed HTML 44. A second set of static HTML follows, proceeded by a length of "0281" in the parsed HTML 44. Finally, the parsed HTML 44 is terminated by a record length of "0000". The PHTML 44 file will typically not physically contain the zero record length indication, but rather this will typically inserted in the end-of-file processing in step 86 in FIG. 3.

FIG. 6 is a block diagram illustrating a General Purpose Computer 20. The General Purpose Computer 20 has a Computer Processor 22, and Memory 24, connected by a Bus 26. Memory 24 is a relatively high speed machine readable medium and includes Volatile Memories such as DRAM, and SRAM, and Non-Volatile Memories such as, ROM, FLASH, EPROM, EEPROM, and bubble memory. Also connected to the Bus are Secondary Storage 30, External Storage 32, output devices such as a monitor 34, input devices such as a keyboard (with mouse) 36, and printers 38. Secondary Storage 30 includes machine-readable media such as hard disk drives, magnetic drum, and bubble memory. External Storage 32 includes machine-readable media such as floppy disks, removable hard drives, magnetic tape, CD-ROM, and even other computers, possibly connected via a communications line 28. The distinction drawn here between Secondary Storage 30 and External Storage 32 is primarily for convenience in describing the invention. As such, it should be appreciated that there is substantial functional overlap between these elements. Computer software such as the parser 42 and user programs can be stored in a Computer Software Storage Medium, such as memory 24, Secondary Storage 30, and External Storage 32. Executable versions of computer software 33, can be read from a Non-Volatile Storage Medium such as External Storage 32, Secondary Storage 30, and Non-Volatile Memory and loaded for execution directly into Volatile Memory, executed directly out of Non-Volatile Memory, or stored on the Secondary Storage 30 prior to loading into Volatile Memory for execution.

Those skilled in the art will recognize that modifications and variations can be made without departing from the spirit of the invention. Therefore, it is intended that this invention encompass all such variations and modifications as fall within the scope of the appended claims.

Claim elements and steps herein have been numbered and/or lettered solely as an aid in readability and understanding. As such, the numbering and/or lettering in itself is not intended to and should not be taken to indicate the ordering of elements and/or steps in the claims.

Claims

What is claimed is:

- 1 1. A method of improving HTML code execution comprising:
 - 2 A) parsing a source HTML file stored in a non-volatile storage
3 medium into a parsed HTML file stored in a non-volatile
4 storage medium, wherein:
5 the parsed HTML file comprises:
6 a set of static HTML records containing static HTML
7 commands, and
8 a set of dynamic HTML records containing dynamic
9 HTML commands,
10 each of the set of static HTML records and each of the
11 set of dynamic HTML records is a variable length
12 record;
13 B) reading the parsed HTML file; and
14 C) processing the parsed HTML file read in step (B), wherein:
15 each of the set of static HTML records is transmitted without
16 interpreting the static HTML commands, and
17 each of the set of dynamic HTML records is interpreted.
- 1 2. The method in claim 1 which further comprises:
 - 2 D) performing a database access in response to interpreting one of the
3 set of dynamic HTML records;
4 E) transmitting data as a result of the database access in step (D).
- 1 3. The method in claim 2 wherein:
 - 2 the database access accesses a hierarchical database.
- 1 4. The method in claim 2 wherein:
 - 2 the database access accesses a relational database.

- 1 5. The method in claim 1 wherein:
2 each of the set of static HTML records and each of the set of dynamic
3 HTML records is a record in a hierarchical database.
- 1 6. The method in claim 1 wherein:
2 each of the set of static HTML records and each of the set of dynamic
3 HTML records is a record in a sequential database.
- 1 7. The method in claim 1 wherein:
2 each of the set of static HTML records and each of the set of dynamic
3 HTML records is a record in a sequential flat file in a variable
4 length record format.
- 1 8. The method in claim 7 wherein:
2 the variable length format comprises:
3 a four character alphanumeric record length, and
4 an ordered set of alphanumeric characters.
- 1 9. The method in claim 7 wherein:
2 an end of the parsed HTML file is indicated by a record length equal
3 to zero.
- 1 10. The method in claim 7 wherein:
2 the sequential flat file utilizes a count-key-data format.
- 1 11. The method in claim 7 wherein:
2 the variable length format comprises:
3 a binary record length, and
4 an ordered set of alphanumeric characters.
- 1 12. The method in claim 7 wherein:
2 the variable length format comprises:
3 a word length,
4 a byte length of a last word, and
5 an ordered set of alphanumeric characters.

- 1 13. The method in claim 7 wherein:
2 the variable length format comprises:
3 a length header comprising:
4 a static HTML record length, and
5 a dynamic HTML record length,
6 a first ordered set of alphanumeric characters containing static
7 HTML commands, and
8 a second ordered set of alphanumeric characters containing
9 dynamic HTML commands.
- 1 14. The method in claim 1 wherein:
2 a one of the set of static HTML records alternates with a one of the
3 set of dynamic HTML records throughout the parsed HTML
4 file.
- 1 15. The method in claim 1 wherein step (A) comprises:
2 1) reading an HTML command as a next HTML command,
3 2) testing whether the next HTML command was a same type as an
4 immediately preceding HTML command,
5 3) writing a current record to the parsed HTML file if the next
6 HTML command was not the same type as the immediately
7 preceding HTML command,
8 4) starting a new record as the current record when the next HTML
9 command was not the same type as the immediately preceding
10 HTML command,
11 5) adding the next HTML command to the current record,
12 6) testing whether there is another HTML command in the source
13 HTML file,
14 7) repeating substeps (1), (2), (3), (4), (5), and (6) as a loop if there is
15 another HTML command in the source HTML file, and
16 8) writing the current record to the parsed HTML file.

1 16. The method in claim 15 wherein step (A) further comprises:
2 testing whether the next HTML command will fit in the current
3 record, and
4 9) writing the current record to the parsed HTML file when the next
5 HTML command will not fit in the current record, and,
6 10) starting a new record as the current record when the next
7 HTML command will not fit in the current record.

1 17. The method in claim 1 wherein:
2 the set of dynamic HTML records contains an SQL command.

1 18. A data processing system for improving HTML code execution
2 comprising:
3 A) means for parsing a source HTML file stored in a non-volatile
4 storage medium into a parsed HTML file stored in a non-
5 volatile storage medium, wherein:
6 the parsed HTML file comprises:
7 a set of static HTML records containing static HTML
8 commands, and
9 a set of dynamic HTML records containing dynamic
10 HTML commands,
11 each of the set of static HTML records and each of the
12 set of dynamic HTML records is a variable length
13 record;
14 B) means for reading the parsed HTML file; and
15 C) means processing the parsed HTML file read in means (B),
16 wherein:
17 each of the set of static HTML records is transmitted without
18 interpreting the static HTML commands, and
19 each of the set of dynamic HTML records is interpreted.

- 1 19. Computer software stored in a computer software storage medium for
2 improving HTML code execution comprising:
3 A) a set of computer instructions stored in a computer software
4 storage medium for parsing a source HTML file stored in a
5 non-volatile storage medium into a parsed HTML file stored in
6 a non-volatile storage medium, wherein:
7 the parsed HTML file comprises:
8 a set of static HTML records containing static HTML
9 commands, and
10 a set of dynamic HTML records containing dynamic
11 HTML commands,
12 each of the set of static HTML records and each of the
13 set of dynamic HTML records is a variable length
14 record;
15 B) a set of computer instructions stored in a computer software
16 storage medium for reading the parsed HTML file; and
17 C) a set of computer instructions stored in a computer software
18 storage medium for processing the parsed HTML file read in
19 set of computer instructions (B), wherein:
20 each of the set of static HTML records is transmitted without
21 interpreting the static HTML commands, and
22 each of the set of dynamic HTML records is interpreted.
- 1 20. The computer software in claim 19 which further comprises:
2 a set of computer instructions stored in a computer software storage
3 medium for performing a database access in response to
4 interpreting one of the set of dynamic HTML records; and
5 E) transmitting data as a result of the database access in set of
6 computer instructions (D).
- 1 21. A computer software storage medium containing the computer
2 software in claim 19.

1/6

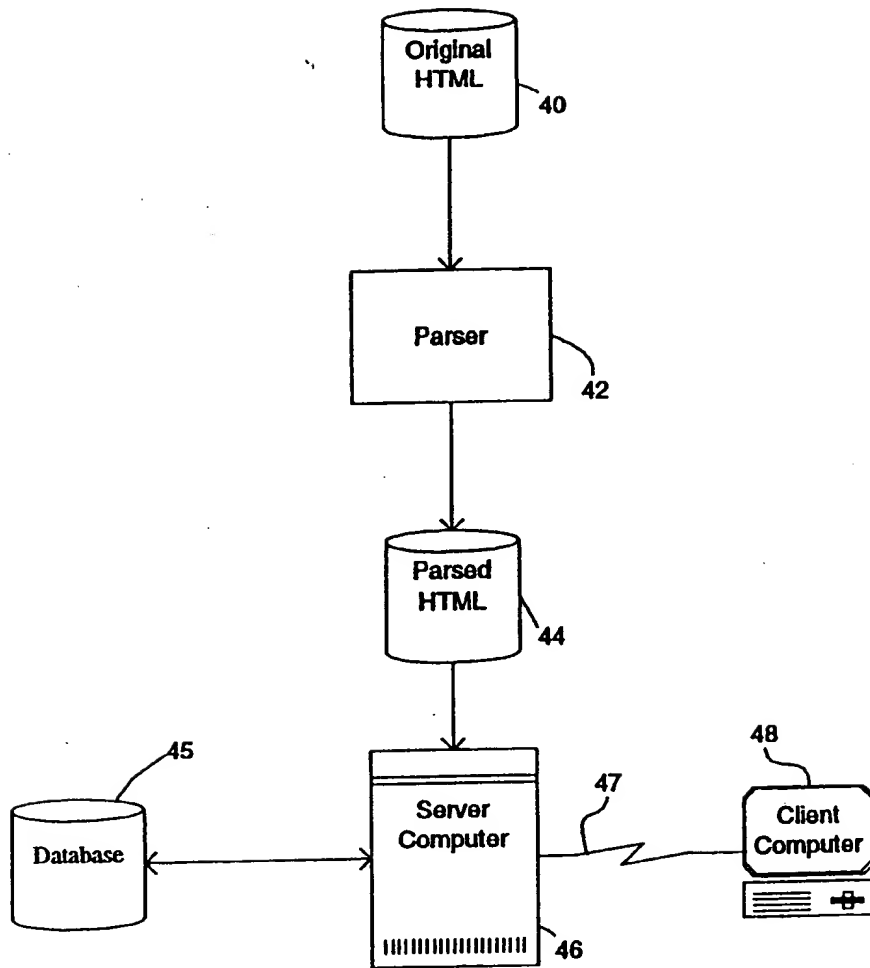


FIG. 1

2/6

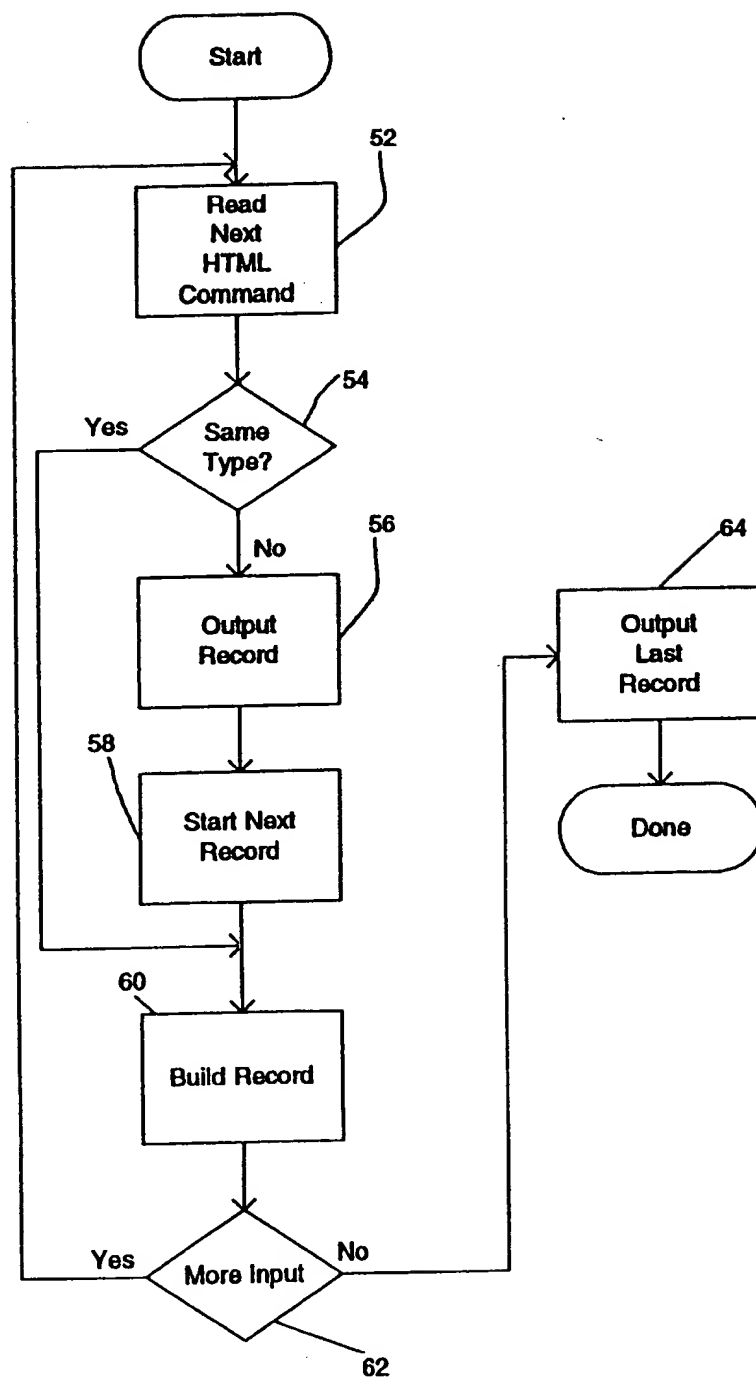


FIG. 2

3/6

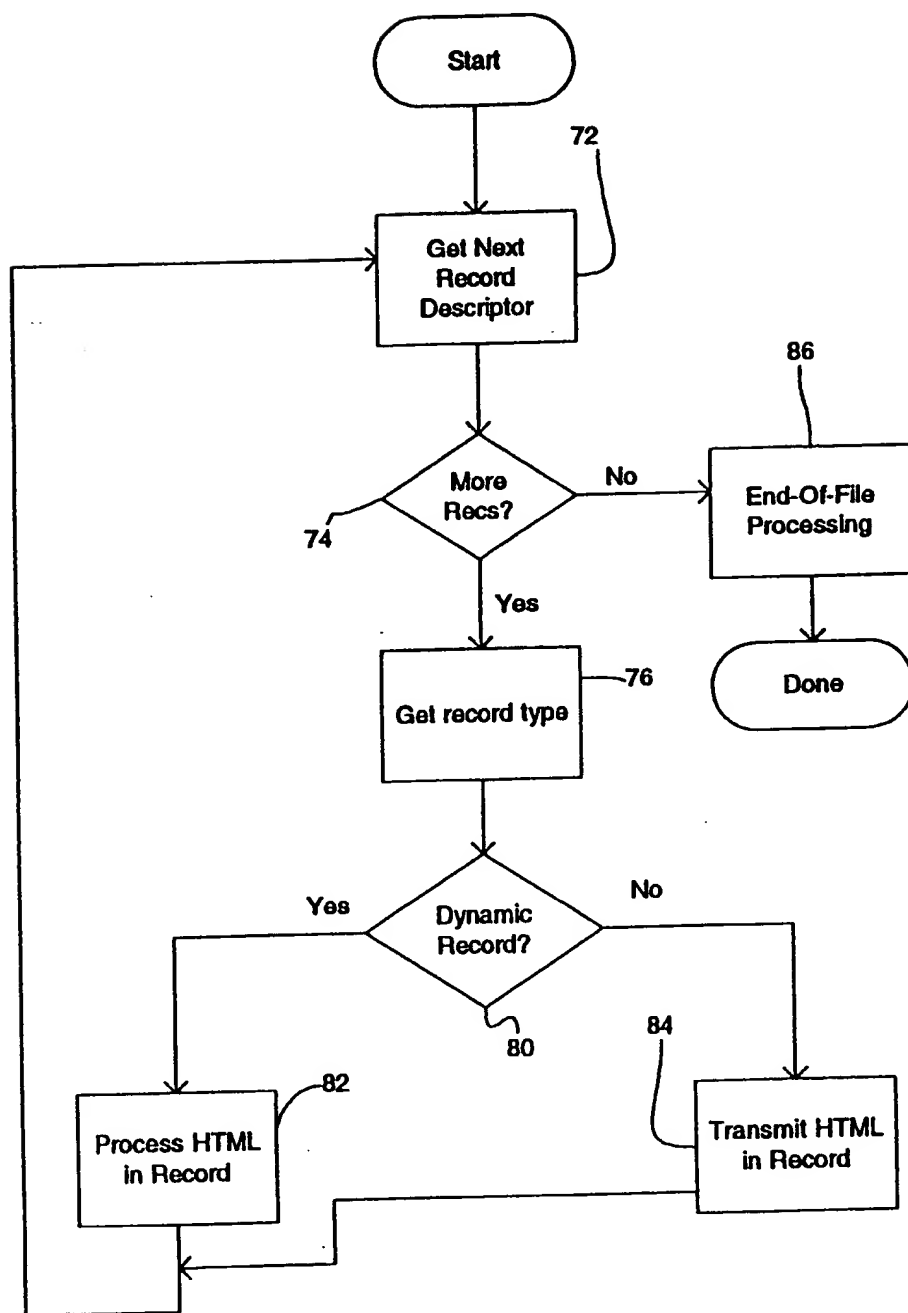


FIG. 3

4/6

Original HTML

```
<HTML>
<HEAD>
<TITLE>SAML DOCUMENT</TITLE>
</HEAD>
<BODY>
<H1> Sample Document for PHTML conversion utility. </H1>
<P>This file serves as an example of a typical document used in Web
    applications. It contains a field that will be replaced dynamically by
    the Web server.</P>
<HR>
<!--SQL: select * from table where field = "value"; -->
<HR>
<P>The preceding HTML comment will be replaced by the Web server with
    the results of the SQL statement. This example assumes that the Web
    server provides access to a database engine and that it uses the HTML
    comment as shown here to flag dynamic content.</P>
</BODY>
<HTML>
```

FIG. 4

5/6

Parsed HTML

0288

<HTML>

<HEAD>

<TITLE>SAMLE DOCUMENT</TITLE>

</HEAD>

<BODY>

<H1> Sample Document for PHTML conversion utility. </H1>

<P>This file serves as an example of a typical document used in Web applications. It contains a field that will be replaced dynamically by the Web server.</P>

<HR>

0059

<!--SQL: select * from table where field = "value"; -->

0281

<HR>

<P>The preceding HTML comment will be replaced by the Web server with the results of the SQL statement. This example assumes that the Web server provides access to a database engine and that it uses the HTML comment as shown here to flag dynamic content.</P>

</BODY>

<HTML>

0000

FIG. 5

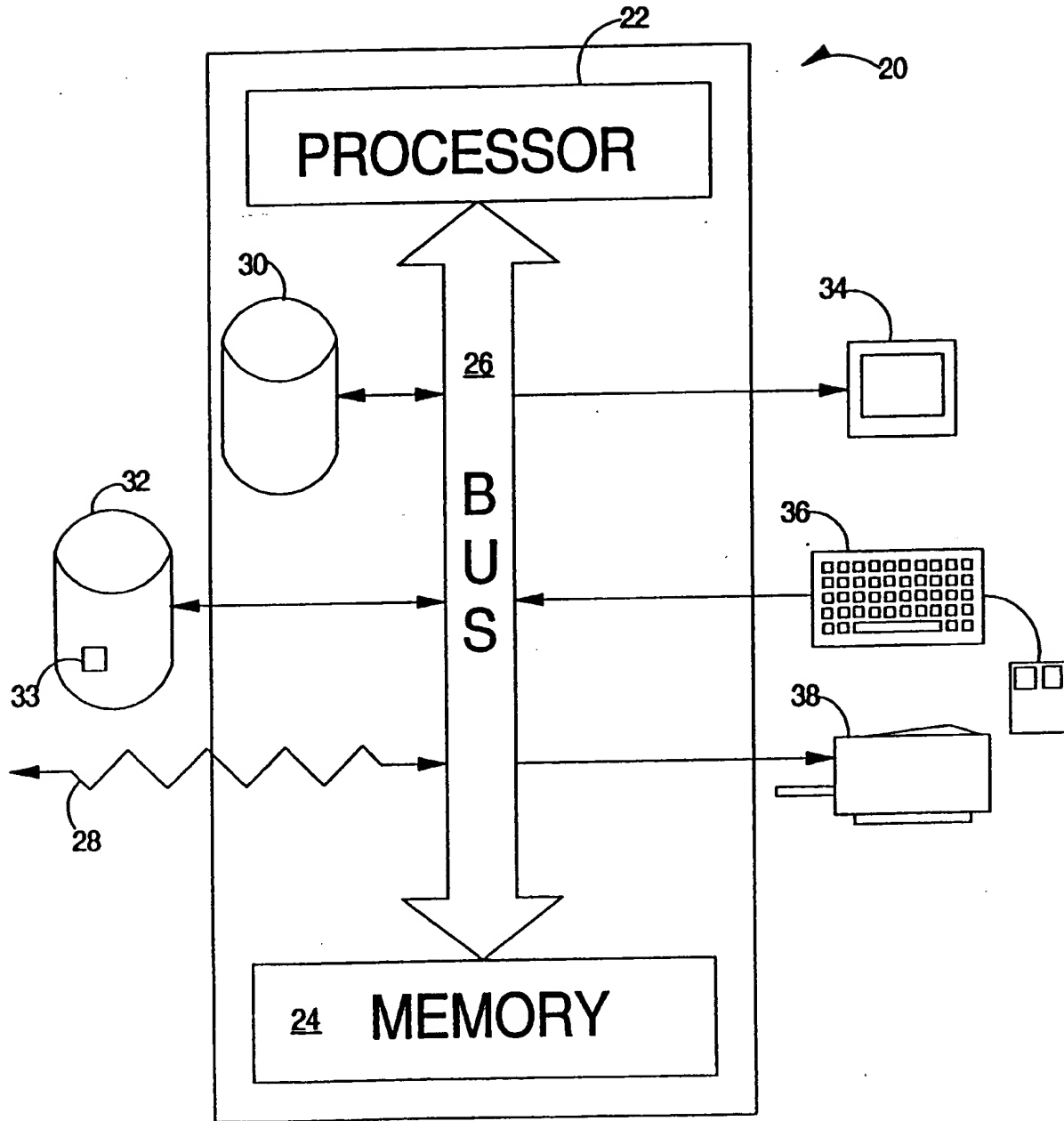


FIG. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/06676

A. CLASSIFICATION OF SUBJECT MATTER IPC(6) : G06F 17/30 US CL : 707/103, 100, 101, 102 According to International Patent Classification (IPC) or to both national classification and IPC		
B. FIELDS SEARCHED Minimum documentation searched (classification system followed by classification symbols) U.S. : 707/103, 100, 101, 102, 10, 3, 4, 5, 1 Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched Electronic data base consulted during the international search (name of data base and, where practicable, search terms used) APS, STN, INSPEC, COMPENDEX		
C. DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	DOUGLIS et al., HPP: HTML macro pre-processing to support dynamic document caching, Proceedings of the USENIX symposium on internet technologies and systems, December 1997, Pages 83-94.[see abstract and entire document]	1-21
Y,P	US 5,844,392 A [PEURACH et al.] 01 December 1998, See figure 2.	1-21
<input type="checkbox"/> Further documents are listed in the continuation of Box C. <input type="checkbox"/> See patent family annex.		
* Special categories of cited documents	*T* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention	
A document defining the general state of the art which is not considered to be of particular relevance	*X* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone	
E earlier document published on or after the international filing date	*Y* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art	
L document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	*G* document member of the same patent family	
-- document referring to an oral disclosure, use, exhibition or other means		
P document published prior to the international filing date but later than the priority date claimed		
Date of the actual completion of the international search 02 JUNE 1999	Date of mailing of the international search report 15 JUN 1999	
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer SANJIV SHAH <i>For [Signature]</i> Telephone No. (703) 305-6355	

T 1/9,K/1

1/9,K/1 (Item 1 from file: 2)

DIALOG(R)File 2:INSPEC

(c) 2006 Institution of Electrical Engineers. All rts. reserv.

04857451 INSPEC Abstract Number: C91032149

Title: Grammar-debugger: a parser for Chinese EFL learners

Author(s): Chen Si-Qing; Xu Luomai

Author Affiliation: Guangzhou Inst. of Foreign Languages, China

Journal: CALICO Journal vol.8, no.2 p.63-75

Publication Date: Dec. 1990 Country of Publication: USA

CODEN: CALJE8 ISSN: 0742-7778

Language: English Document Type: Journal Paper (JP)

Treatment: Practical (P)

Abstract: The paper presents a parser which is an extension of the deterministic model of natural language processing. The **parser** , named **Grammar** -Debugger, is made more powerful than the earlier models by a modified lookback mechanism and an additional error detecting and tolerating mechanism. The error detection and toleration capability is achieved by a minor modification of the grammar rules which are used to parse grammatical sentences, thus the generality of the grammar rules is preversed. The targeted application of the parser is in the field of CALL.

(6 Refs)

Subfile: C

Descriptors: computer aided instruction; grammars; linguistics; natural languages

Identifiers: English as foreign language; parser; Chinese EFL learners; natural language processing; Grammar-Debugger; lookback mechanism; error detecting; toleration capability; CALL

Class Codes: C7810C (Computer-aided instruction); C4210 (Formal logic); C6180N (Natural language processing); C7820 (Humanities)

...Abstract: a parser which is an extension of the deterministic model of natural language processing. The **parser** , named **Grammar** -Debugger, is made more powerful than the earlier models by a modified lookback mechanism and...

?